

FSMLabs RTLinux technology for hard realtime

Nicholas Mc Guire
der.herr@hofr.at
FSMLabs Data GmbH - AUSTRIA

Schedule

- Hard, Soft and non-realtime ?
- Who needs hard realtime ?
- Introduction to RTLinux
- Availability

Slide 1

Slide 2

Hard Realtime

- Guaranteed worst case jitter
- High resolution timer functions
- Guaranteed worst case interrupt response times
- Not one event will be missed
- the system is deterministic

Slide 3

Soft Realtime

- Low average jitter
- High resolution timer functions
- Low average interrupt response times
- Events may be missed
- The system is statistically predictable

Slide 4

Non-Realtime

- Timing functions have low accuracy
- There is absolutely no guarantee for worst or even average case
- All timing related functions are load dependant
- The system is optimized for overall efficiency
- A non-realtime system is not deterministic at all

Slide 5

Typical Timings

Timing requirements in a normal PC, most commonly managed by specialized hardware.

- 1.5 microseconds between frames on a gigabit Ethernet
- Frame intervals of 14 milliseconds on a 70Hz video display
- 1.2MB/s data to the CD-burner load independent
- Video Conferencing requiring updates at 20Hz minimum

Slide 6

Non-realtime might do

You can do without hard realtime systems if

- You are happy with good average performance
- Missing events is not a major problem as long as most are handled correctly
- You don't have any timing constraints in your system
- Time-resolution of the system in the second or millisecond range is enough

Slide 7

Hard Realtime is a requirement

You need hard realtime for systems if any of the following applies

- Timer resolution must be below the millisecond level
- Event resolution must be at or below the millisecond range
- You must handle every event of a specific type (even if it's slow this will require hard realtime capabilities)
- Your system must behave deterministic

Slide 8

The RTOS design dilemma

The requirement of strict time management leads to contradictory requirements

- Only a small simple OS can be deterministic
- Users want TCP/IP, GUIs and compatibility to normal desk-top systems

Slide 9

Make a general purpos OS to and RTOS ?

Why can't one simply add realtime capabilities by modifying a general purpose OS and making it fully preemptive ?

- General Purpose Operating Systems are event driven not time triggered.
- Modifying all applications to be preemptive would be very costly and error prone
- Maintenance cost of such a system would be high since it could not resort to available software packages

Expand an RTOS to a general purpos OS ?

Could one simply add the additional capabilities like TCP/IP and X11 to a small RTOS ?

A few problems would arise

- All core capabilities must be preemptive to guarantee worst case timing delays
- Priority inversion as in regular OS's may not occur
- Many mechanisms for efficiency become problematic (caching, queuing etc)
- Available software packages would need to be modified and this would result in high maintenance effort and cost

Slide 10

FSMLabs RTLinux

By splitting the problem RTLinux can provide a solution to this dilemma

- RTLinux is a hard realtime operating system
- RTLinux follows the POSIX PE51 1001.13 RT profile
- It's a small compact layer "below" Linux
- RTLinux only handles one resource : timing

Regular Linux applications don't need to be aware of these modifications if they don't need hard realtime capabilities.

Slide 11

Slide 12

A POSIX compliant RT-process

- A RT kernel provides the scheduling instance as the "main" routine
- An Application Signal Handler provides IPC by signals
- The Application Thread provides the context of the RT-task

Slide 13

What happens to Linux in RTLinux

- Linux is only a thread managed by the realtime scheduler
- RTLinux runs Linux as its lowest priority thread "idle task"
- Linux can communicate with the "realtime side" via signals and FIFO's as well as through shared memory

Slide 14

What does RTLinux supply to Linux ?

- High resolution timing functions
- Advanced interrupt control functions
- FIFO's from the realtime side to the non-realtime side
- Shared memory accessible from Linux and RT-processes
- Signals between RT and non-RT processes

Slide 15

What is the RTLinux Patent ?

The RTLinux Patent covered by US Patent: 5,995,745

- Put an emulated interrupt controller in a general purpose OS
- Prevent interrupt disabling by the general purpose OS
- Run the general purpose OS as the lowest priority thread

Slide 16

Interrupt Emulation

- Linux cli does "clear soft enable bit"
- Linux sti does "set soft enable bit"
- On occurrence of an interrupt

```
    if there is an RT handler: call it
    if there is a linux handler AND RT is idle
                                AND soft interrupts are enabled
                                call the Linux handler
    else mark the interrupt pending
```

Slide 17

License Issues

- Basic Technology related to RTLinux will be kept conforming to the GPL
- Customer specific developments of drivers or special purpose extensions may be covered under other terms.
- RTLinux is royalty free

Slide 18

What FSMLabs provides

- Core RT-kernel development
- Services for GPL Software related to RTLinux
- Customer specific GPL or non-GPL Software

Slide 19

Where to get it

RTLinux is available from FSMLabs directly or you can download it from the Internet

- FSMLabs
<http://www.fsmlabs.com>
<ftp://ftp.fsmlabs.com>
- RTLinux.org
<http://www.rtlinux.org>
<ftp://ftp.rtlinux.org>
- And a bunch of mirror sites around the globe